REMARKS

Reconsideration of the pending application is respectfully requested on the basis of the following particulars:

Rejection of claims 1-6 and 8-12 under 35 U.S.C. § 103(a)

Claims 1-5 and 8-12 presently stand rejected as being unpatentable over Peyret et al (U.S. 5,923,884) in view of Raith (U.S. 6,493,550) and Wambach et al (U.S. 6,330,648), and claim 6 is rejected as being unpatentable over Peyret, Raith, and Wambach in further view of Carper (U.S. 6,256,690). These rejections are respectfully traversed for the following reasons.

Claims 1, 4, 8, and 12 are currently amended, as presently shown above in the "Amendment to the Claims." In the amendment of claims 1, 4, 8, and 12, each of these independent claims points out that the loader interface transfers the management of an assigned address space completely to a load application after the load application is assigned to the assigned address space, such that the loader interface has no further influence on or access to the assigned address space.

It is respectfully submitted that the examiner's proposed combination of Peyret, Raith, and Wambach fails to disclose or suggest a loader interface that transfers the management of an assigned address space completely to a load application after the load application is assigned to the assigned address space, such that the loader interface has no further influence on or access to the assigned address space.

With respect to claim 1, it is further submitted that the cited references fail to disclose or suggest that a loader interface loads a second loader (the load application configured to load at least one application program into the assigned address space according to claim 1).

Accordingly, the combination of Peyret, Raith, and Wambach fails to establish a prima facie case of obviousness against the pending claims.

The rejections of the present Office Action essentially repeat the rejections of the previous Office Action (mailed on September 15, 2005), except the examiner now asserts that "furthermore, Microsoft provides a "loader program" (see Control Panel, 'Add or Remove Programs') which provides loading functionality and memory area protection."

In response to Applicant's arguments set forth in response to the previous Office Action (the response filed on January 17, 2006), the examiner asserts that "paramount in the examiner's rejection is the fact that the independent claims are written in broad language and to not specifically define components and/or operations that are novel," and "the claims appear to recite that which is known in the art [...]."

The examiner is reminded that "most if not all inventions arise from a combination of old elements." *In re Kotzab*, 217 F.3d 1365, 1369 (Fed. Cir. 2000). "Thus, every element of a claimed invention may often be found in the prior art." *id.* "However, identification in the prior art of each individual part claimed is insufficient to defeat patentability of the whole claimed invention." *id.*

Accordingly, the examiner's mere assertions that the claims do not "specifically define components and/or operations that are novel" and that "the claims appear to recite that which is known in the art" do not, even assuming, *arguendo*, that they are accurate, form a prima facie case of obviousness of the present claims. "To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations." (MPEP 2143).

It is respectfully submitted that the references cited by the examiner fail to teach or suggest all the claim limitations set forth by the claims of the present application.

According to each of the independent claims, a loader interface of the present invention loads a load application, allots a part of free memory space to the load

application as an assigned address space, and then assigns management of the assigned address space completely to the load application, such that the loader interface has no further access to the assigned address space.

None of the cited references, either individually or in combination, teach or suggest such an arrangement.

The examiner again notes that "[Peyret] is silent on equipped with a communication device and wherein the loader interface cannot access an assigned address space after a load application is assigned to the assigned address space."

As pointed out in response to the previous Office Action, Raith is entirely silent on the use of any type of loader or any other means of loading applications onto a smart card, and therefore fails to supplement the shortcomings of Peyret regarding the loader interface and load application.

The examiner again turns to Wambach. However, Wambach does not disclose a method or device for preventing memory access, since even while a region of memory subject to Wambach's anti-overwrite device is prevented from being written to, access to the region of memory is still allowed in the form of memory reads. Thus, according to the anti-overwrite device according to Wambach, it appears that all programs or devices in a given system are still able to access the write-protected memory region to read the memory.

Providing memory protection (against ***any and all*** write operations), as taught by Wambach, is different from an operation of a first loader interface giving up any and all access to an assigned address space, while a loaded load application is given full access to the same assigned address space.

Wambach fails to teach or suggest how one program could give up all access to the assigned address space, while at the same time another program has full access for management and use of the assigned address space. Instead, Wambach simply teaches a memory protection (anti-overwrite) scheme wherein a section of memory may be protected against memory writes by ***all*** potential write accessors.

The examiner seeks to bolster his position with a reference to Microsoft Windows. Without reference to documentary evidence, the examiner nonetheless comments that "systems such as Microsoft Windows have memory areas that programs are downloaded to which are protected from being overwritten by user data, as described by Wambach," and that "furthermore, Microsoft provides a 'loader program' (see Control Panel, 'Add or Remove Programs') which provides loading functionality and memory area protection."

The examiner's assertions with respect to Microsoft Windows do not provide any teaching or suggestion of a loader interface that (upon loading a load application) gives up any and all access to an assigned address space, while a loaded load application is given full access to the same assigned address space.

While an operating system such as Microsoft Windows may indeed "have memory areas that programs are downloaded to which are protected from being overwritten by user data," it does not follow that the operating system ever relinquishes management authority over such a memory area. On the contrary, it is respectfully submitted that memory management is typically an essential feature of such an operating system, and that in an operating system such as Microsoft Windows it is the operating system itself that performs memory management for loaded programs, through functions and services that are provided by the operating system itself.

The examiner's own assertion that "Microsoft provides a 'loader program' (see Control Panel, 'Add or Remove Programs')" contradicts the idea that a loader interface transfers management of an assigned address space completely, such that the loader interface has no further access to the assigned address space, since a "Remove Programs" operation performed by Microsoft Windows inherently requires that at least the "Remove Programs" component of Microsoft Windows retains an authority over a memory space which a loaded program cannot trump.

Applicant also notes that various other functions, such as disk defragmentation programs, virus scanning software (that scan both RAM and mass-storage devices after a program has been loaded), memory testing programs, and numerous other examples,

access memory areas and devices even while the memory areas and devices are occupied for the use of loaded programs.

It is respectfully submitted that, for at least these reasons, the cited references, either individually or in combination, fail to disclose or suggest all the claim limitations set forth by the claims of the present application. Accordingly, it is respectfully submitted that claims 1-12 are allowable, and withdrawal of these rejections is respectfully requested.

Conclusion

In view of the amendments to the claims, and in further view of the foregoing remarks, it is respectfully submitted that the application is in condition for allowance. Accordingly, it is requested that claims 1-12 be allowed and the application be passed to issue.

If any issues remain that may be resolved by a telephone or facsimile communication with the Applicant's attorney, the Examiner is invited to contact the undersigned at the numbers shown.

Respectfully submitted,

BACON & THOMAS, PLLC
625 Slaters Lane, Fourth Floor
Alexandria, Virginia 22314-1176
Phone: (703) 683-0500

Date: May 1, 2006

JUSTIN J. CASSELL
Attorney for Applicant
Registration No. 46,205